Journal Title, 2022, pp. 1–9

doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

Towards improving full-length ribosome density prediction by bridging sequence and graph-based representations

Mohan Vamsi Nallapareddy,[1,*] Francesco Craighero,[1] Cédric Gobet,[2] Felix Naef[2] and Pierre Vandergheynst[1]

[1]LTS2 Signal Processing Laboratory, IEL, STI, École Polytechnique Fédérale de Lausanne, Rte Cantonale, 1015, Vaud, Vaud and [2]UPNAE Laboratory of Computational and Systems Biology, IBI, SV, École Polytechnique Fédérale de Lausanne, Rte Cantonale, 1015, Vaud, Switzerland

[*]Corresponding author: vamsi.nallapareddy@epfl.ch

FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Translation elongation plays an important role in regulating protein concentrations in the cell, and dysregulation of this process has been linked to several human diseases. In this study, we use data from ribo-seq experiments to model ribosome dwell times, and in turn, predict the speed of translation. The proposed method, RiboGL, combines graph and recurrent neural networks to account for both graph and sequence-based features. The model takes a mixed graph representing the secondary structure of the mRNA sequence as input, which incorporates both sequence and structure codon neighbors. In our experiments, RiboGL greatly outperforms the state-of-the-art RiboMIMO model for ribosome density prediction. We also conduct multiple ablation studies to justify the design choices made in building the pipeline. Additionally, we use gradient-based interpretability to understand how the codon context and the structural neighbors affect the ribosome dwell time at the A site. By individually analyzing the genes in the dataset, we elucidate how structure neighbors could also potentially play a role in defining the ribosome dwell times. Importantly, since structure neighbors can be far away in the sequence, a recurrent model alone could not easily extract this information. This study lays the foundation for understanding how the mRNA secondary structure can be exploited for dwell time prediction, and how in the future other graph modalities such as features from the nascent polypeptide can be used to further our understanding.

Key words: Graph Neural Networks, Ribosome Density Prediction, Graph Interpretability

Introduction

Translation, the process in which RNA nucleotide triplets are encoded into amino acids to build proteins, plays a vital role in cell function. Translational control allows for rapid changes in the concentrations of encoded proteins in the cells. Thus, translational control plays an important role in maintaining homeostasis, and in modulating more permanent changes in cell fate or physiology [1]. Errors in the translation machinery are the cause of a variety of human diseases, including certain cancers and metabolic disorders. Dysregulation of signaling pathways that control cell growth and proliferation can lead to cancers, and these pathways also affect translation. Cancer is also associated with abnormal changes in the amounts of tRNAs, translation regulatory factors, and initiation factors. In particular, translation elongation has emerged as an important process that is often dysregulated in these diseases [2].

Ribo-seq is a technique to obtain ribosome read count values at codon resolution. Learning how one can predict these values can help us understand important translation-related phenomena such as ribosome stalling, ribosome collisions, and synonymous codon bias. This could also potentially elucidate the underlying mechanisms of various metabolic diseases. Recently, we have seen the application of machine learning models to predict ribosome read count values using information from the mRNA sequence [3, 4, 5, 6, 7, 8].

In this study, we model the mRNA sequence as a graph using its predicted secondary structure and design a graph-based approach to predict full-length ribosome densities (see **Figure 1**). To the best of our knowledge, the proposed approach, named RiboGL, is the first one that leverages the graph nature of the mRNA secondary structure. This approach is different from the ones proposed in the literature, where the mRNA is modeled in terms of a sequence [8] or as a codon-context window [5]. More in detail, the mRNA is encoded as a *mixed graph*, where codon sequence neighbors within the mRNA are represented by directed edges, following the ribosome movement direction, while undirected edges correspond to structural neighbors in the secondary structure. Additionally,

© The Author 2022. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

RiboGL is the first one to learn the embeddings of the codons, as opposed to using one-hot encodings as the features.

Previous approaches for predicting the ribosome density at a specific position exploited only the information of neighboring codons [8]. By taking into account the secondary structure, the model can exploit structural neighbors to capture codons at a much greater distance. Indeed, learning such interactions with a small and noisy dataset might be prohibitive. This is also particularly relevant since sequence-based models like Recurrent Neural Networks (RNNs) might fail in capturing long-term dependencies due to vanishing gradients [9]. Conversely, Graph Neural Networks (GNNs) can easily handle the structural neighbors by accepting the secondary structure as input, but might fail at capturing information from distant nodes [10]. Consequently, we defined RiboGL as a GNN graph processing block, to extract the information from the structural neighbors, followed by a sequence processing block, overcoming the limitations of GNNs. To fully exploit the directionality information encoded in the mixed graph, we chose the recently introduced Directed Graph Neural Network (Dir-GNN) [11]. Moreover, as a graph convolution, we employed the TransformerConv (TrConv) [12], which in our experiments reached the highest performance. The node features learned by the GNN are then processed by the sequence block composed of a Bi-Directional Long Short Term Memory (LSTM) [13] model. **Figure 1** outlines the summarized RiboGL pipeline.

Modeling ribosome density prediction with both sequence and graph features allows us to learn new codon interactions, that can be extracted with post-hoc interpretability techniques [14]. Indeed, while with sequence-based models we can estimate only codon-wise feature importance, graph inputs enable us to extract the flow of information between neighbors through edge importance.

Contributions.

In section 4.1, we show that the proposed RiboGL model outperforms the state-of-the-art RiboMIMO [8] model by ∼19%. In section 4.2 we conduct multiple ablation studies to justify our design choices for RiboGL. Lastly, in section 4.3 we show the applications of RiboGL for interpretability, where we extract the contribution of both the codons and the edges of the secondary structure graph. To reproduce the experiments, refer to `https://github.com/vam-sin/ribogl/`.

Related Work on Ribosome Density Prediction

Earlier approaches to predict ribosome density, namely riboShape [3] and RUST [4], employed linear models and codon-wise statistics to infer ribo-seq densities. To denoise the dwell times, the former employed wavelets and kernel smoothing, while the latter binarized values to reduce the impact of outliers. More recent approaches exploited deep learning to handle the complexity of the ribo-seq data. ROSE [5] trained a Convolutional Neural Network (CNN) as a binary classifier to detect stalling events. Iχnos [6] and Riboexp [7] used a Feedforward and a Recurrent Neural Network (FNN and RNN), respectively, to predict each codon density using both the information about its neighbors and their RNA folding energy. While previous approaches took into consideration a window or context around the target codon, the current state-of-the-art, RiboMIMO [8], is trained to predict the whole density profile of a transcript. Similar to ROSE, RiboMIMO considers a simplified classification task to predict dwell times, while also adding another regression loss on the normalized counts. In **Table A.1**, we summarized the characteristics of each approach.

Methods

Mouse Liver Dataset

Mouse liver ribosome profiling data from [16], available in the Gene Expression Omnibus (GEO) database under accession number GSE73553, was used in this study. This data was pre-processed through the initial steps of our "Ribo-DT" snakemake pipeline[1], with slight modifications, to generate position-specific ribosome A-site coordinates on the transcriptome, as outlined in [17]. Specifically, mouse genome sequences (GRCm38/mm10), and transcript annotations were downloaded from ENSEMBL (Release 95). Sequence Read Archive (SRA) files were retrieved using the GEO accession number and then converted into FASTQ format. These files were then aligned to the mouse genome using STAR with inline adapter clipping (TGGAATTCTCGGGTGCCAAGG). The resulting BAM files were indexed. Size-dependent A-site positions were computed using a pile-up of 5'-end read density at the start codon for each read size and frame. Unique mapping reads of size between 26 and 35 nucleotides and up to one mismatch were included in the analysis. Read counts and coding DNA sequence (CDS) positions were retrieved, with the A-site offset adjusted accordingly.

The codon-level ribosome counts were normalized by the gene average, similar to previous approaches [8]. Moreover, we kept only sequences with coverage, defined as the percentage of non-zero and non-NaN annotations, greater than 30%. The ranges in the annotations were reduced by applying a log1p function. The resulting dataset consisted of 6,188 genes, with 20% left out as a test set. More details on our preprocessing are available in section A.1, while the coverage density was reported in A.1.

Secondary Structure Prediction

The secondary structure of the mRNA sequence was used as the input to the RiboGL model. The graphs of the mRNA sequences were obtained by using the ViennaRNA [15] module with the Minimum Free Energy (MFE) approach [18] (`RNA.fold` from the ViennaRNA Python API). An example structure is reported in **figure 1 (A)**. This module results in the dot-bracket secondary structure for the mRNA, which is then converted into an adjacency matrix. The created adjacency matrix would correspond to the nucleotides, this is then pooled to create a codon-level adjacency matrix which is used as the input to the RiboGL model. An example sequence and the corresponding adjacency matrix have been mentioned in **figures 1 (B) and (C)**.

Graph Neural Networks (GNNs)

Graph Neural Networks are a special class of neural networks designed to process graph-based inputs. Consider a graph $G = (V, E)$ where node set V represents the n nodes and edge set E represents the m edges. The adjacency matrix $\mathbf{A} \in (0,1)^{nxn}$ is populated based on the directionality of the edges. If there is a directional edge from node i to node j, then $a_{j,i}$ is set to 1, and $a_{i,j}$ is set to 0. But if there is an undirected edge between

[1] `https://github.com/cgob/codonDT_snakemake`

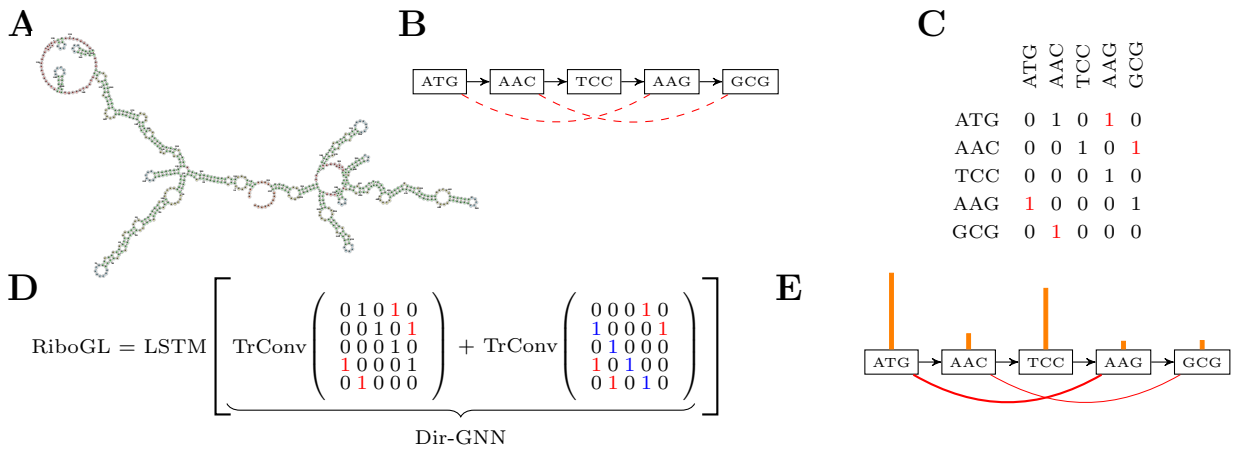

Fig. 1. Overview of the RiboGL pipeline. (A) The mRNA secondary structure of the example gene "Mrpl11" as predicted by the ViennaRNA web server [15]. (B) Graphical depiction of how the edges are modeled using an example sequence. The codons that are sequence neighbors are connected to each other using directed edges (in black), and codons that are structure neighbors are connected to each other using undirected edges (in red). (C) The adjacency matrix of the example graph outlined in (B), with undirected edges in red. (D) The graph processing equation for the example sequence. Following the definition of Dir-GNN, two Transformer Convolutions are applied individually, one to the original adjacency matrix and the other to the transposed adjacency matrix, and the outputs are summed. Then, an LSTM is applied to the learned node features. (E) Interpretability analysis on the example sequence, this can help understand the attributions of all the codons and the edges in the graph. The height of the orange bars indicates the magnitude of contribution from the codons, and the intensity of red on the edges indicates the magnitude of contribution from the edges.

nodes i and j, then both $a_{i,j}$ and $a_{j,i}$ are set to 1. GNNs follow a Message Passing Neural Network (MPNN) paradigm where the information from neighboring nodes is used to compute new node-level features in every pass of the network. The node features of node i are represented as x_i. The kth pass of the MPNN, is defined by aggregation functions ($AGG^{(k)}$) and combination functions ($COM^{(k)}$) which are used iteratively to compute embeddings x_i^k for node i based on messages m_i^k that contain information from its neighbours.

$$\begin{aligned} m_i^k &= AGG^{(k)}\left(\left\{\left(x_j^{k-1}, x_i^{k-1}\right) \colon (i,j) \in E\right\}\right) \\ x_i^k &= COM^{(k)}\left(x_i^{k-1}, m_i^k\right) \end{aligned} \tag{1}$$

The general working of the MPNN has been outlined using equation 1. The $AGG^{(k)}$ and $COM^{(k)}$ vary with different implementations of the MPNN and can result in different architectures such as the Graph Convolutional Network (GCN) [19], and the Graph Attention Network (GAT) [20].

In this study, we intend to explore how the neighborhood of the codon A site affects the ribosome density. Studying the influence of all the edges and nodes in the graph would help us understand more about the features of the codon neighborhood that affect the speed of translation. There have been several graph explainability algorithms that have been suggested previously [21], but to the best of our knowledge, they haven't been applied to the setting of ribosome dwell time prediction.

RiboGL

The RiboGL model is composed of a learnable embedding of size 128, followed by two blocks in series:

1. **Graph Processing Block (RiboGL-GNN):** The purpose of the graph processing block is to exploit the graph structure of the mRNA, and it consists of four graph convolution layers with 256, 128, 128, and 64 channels respectively. These graph convolution layers use the TrConv algorithm [12] to process their inputs. This algorithm learns two separate weight matrices W_1 and W_2, for node i and the neighbors of node i respectively. Additionally, the weights for the neighborhood feature matrices are derived using the attention mechanism. The working of this has been outlined using equation 2. These convolutions were modified to exploit the directionality, using the Directed-Graph Neural Networks (Dir-GNN) [11] approach. This allows us to add additional information about the direction of the translating ribosome to the input secondary structure graph. The implementation consists of the convolution operation applied twice, once on the original adjacency matrix, and again on the transposed adjacency matrix (see **Figure 1 (D)**). The outputs from these convolutions are combined to obtain the final output. The working of this directed convolution has been outlined in equation 3, where E^T represents the flipped edge list of the graph and $V = [x_0^K, \ldots, x_n^K]$ contains either the previous layer node features or the input features when $K = 0$.

$$\begin{aligned} x_i' &= W_1 x_i + \sum_{j \in N(i)} \alpha_{i,j} W_2 x_j \\ \alpha_{i,j} &= \text{softmax}\left(\frac{(W_3 x_i)^T (W_4 x_j)}{\sqrt{d}}\right) \end{aligned} \tag{2}$$

$$\begin{aligned} RiboGL - GNN &= [x_0^{K+1}, \ldots, x_n^{K+1}] \\ &= \text{TrConv}(V, E) + \text{TrConv}(V, E^T) \end{aligned} \tag{3}$$

The outputs from each layer of this block are concatenated together to construct the final output. This is conducted using the Jumping Knowledge module [22]. This helps retain the information from all the layers of the graph processing block.

2. **Sequence Processing Block (RiboGL-LSTM):** This consists of a 4-layer bi-directional long-short term memory (BiLSTM) model with 128 nodes each. This block was added so that we could potentially augment information by processing the mRNA in terms of a sequence as well.

RiboGL Variant	Input Graph Structure	Dir-GNN [11]
USeq	useq	No
USeq+	useq, struc	No
DirSeq	dseq	Yes
DirSeq+	dseq, struc	Yes

Table 1. Variations of the RiboGL model, with different input graph structure and convolution types Input Graph Structure: undirected sequence edges (useq), directed sequence edges that incorporate the direction of the translating ribosome from 5' to 3' (dseq), and undirected structure edges (struc).

Equation 4 outlines the working of the RiboGL-LSTM, where x_i^K represents the output embeddings of node i from the final K^{th} layer.

$$RiboGL - LSTM = LSTM([x_0^K, \ldots, x_n^K]) \quad (4)$$

The overview of the RiboGL model outlining the two components, and the directed nature of the convolutions can be found in **Figure 1 (D)**. We have tested out four variants of the RiboGL model, USeq, USeq+, DirSeq, and DirSeq+, the properties of these models have been outlined in **Table 1**. We start with the simplest version, the USeq model, which only uses the undirected sequence as the input, and add additional features to design the other three variations. Additionally, in the undirected variants, USeq, and USeq+, the graph convolutions are only applied only once on the original adjacency matrix.

The proposed RiboGL model is the DirSeq+ variant. For each of the nodes in the graph, the model uses learned embeddings from the one-hot encoding of the codon. To train RiboGL, a node-level regression task was performed. This way for each codon the model predicted a normalized ribosome density value, which would then be concatenated together to obtain a ribosome profile for the entire gene. The training process was regularized using GraphNorm [23] and Dropout [24] layers. The losses and metrics used in this training process have been outlined in appendix section A.2, and the model hyperparameters are mentioned in section A.3.

Captum Graph Explainer

In order to explain the results on various genes, we studied what codons in the graph affected the peaks in the ground truth ribosome profile. For each gene, the top 10 codons in the ground truth ribosome profile in terms of their dwell time magnitude were chosen and were perturbed using a Captum [25] based explainer employing the "Input X Gradients" [14] algorithm. This method provided attribution values for the chosen peaks with respect to all the nodes, and the edges in the secondary structure graph. This would allow us to understand the relationships of the codons with the sequence and structure neighborhoods. An example of the interpretability output from the Captum Graph Explainer can be found in **Figure 1 (E)**. The height of the orange bars represents the magnitude of the contribution from the codon, and the intensity of the red color for the edges represents the magnitude of their contribution.

Model	PCC ↑	MAE ↓
RiboMIMO	0.4459 ± 0.00872	0.64931 ± 0.01380
RiboGL - GNN	0.4744 ± 0.0043	0.03969 ± 0.00081
RiboGL - LSTM	0.6356 ± 0.0038	0.03732 ± 0.00070
RiboGL - DirSeq+	**0.6378 ± 0.0039**	**0.03715 ± 0.00066**

Table 2. RiboGL comparison with the state-of-the-art RiboMIMO model and the individual components of RiboGL, RiboGL-GNN, and RiboGL-LSTM. The performances mentioned are the mean and 95% confidence intervals derived by conducting bootstrapping on the testing set of the mouse liver dataset, refer to appendix A.4 for more information on the bootstrapping process

Model	PCC ↑	MAE ↓
RiboGL - USeq	0.6318 ± 0.0035	0.03740 ± 0.00075
RiboGL - USeq+	0.6279 ± 0.0035	0.03754 ± 0.00073
RiboGL - DirSeq	**0.6435 ± 0.0038**	0.03718 ± 0.00072
RiboGL - DirSeq+	0.6378 ± 0.0039	**0.03715 ± 0.00066**

Table 3. RiboGL ablation study to determine the optimal input graph structure. Comparison of model performances of the four different variants of the RiboGL model, which was conducted to identify the usefulness of incorporating graph directionality and additional structural neighbors from the mRNA secondary structure.

Analysis of the Results

Comparing sequence and graph-based representations

The RiboGL model has a performance of 0.6378 in terms of the mean Pearson Correlation Coefficient (PCC) on the mouse liver dataset (section 3.1). This is ∼19% greater than the performance of the state-of-the-art RiboMIMO model which was re-trained and then tested on this dataset. Therefore, the proposed RiboGL model significantly outperforms the state-of-the-art model. This comparison has been outlined in **Table 2**. Additionally, we compare the RiboGL model with its individual components, the RiboGL-GNN and RiboGL-LSTM. The performance of the full RiboGL model was ∼16% greater than that of the RiboGL-GNN model, but only slightly better than the RiboGL-LSTM model. Our hypothesis is that the GNN alone is unable to exploit the full codon-context around the A-site due to the limitations of these models with long-range dependencies [10]. The clear difference between the two RNNs, RiboMIMO and RiboGL-LSTM, is due to the fact that RiboMIMO does not have a learnable embedding like RiboGL and its components. Lastly, the small difference between RiboGL-LSTM and RiboGL could be due to the fact that structural neighbors are helping the model only in small regions of each gene, and such improvement is not immediately reflected by our dataset-wise metrics.

Effect of Graph Directionality and Structural Neighbors

To understand the importance of the different design aspects in creating the RiboGL model, four different variants of it, DirSeq+, DirSeq, USeq+, and USeq were tested. These four models have been compared in **Table 3**.

Taking into account the directionality of the input mixed graph through Dir-GNN improves the performance, since RiboGL DirSeq and DirSeq+ are better than their fully undirected variants Useq and Useq+. On the other hand, including structural neighbors does result in a slight worsening

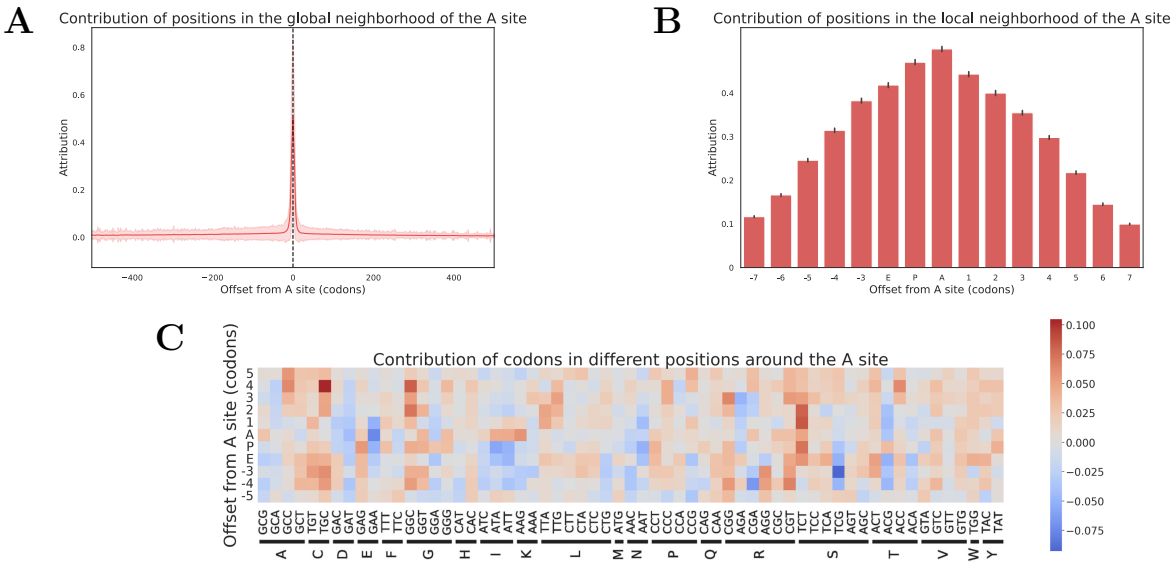

Fig. 2. Attribution analysis conducted using the Captum Graph Explainer in order to understand the average contribution of codons to the ribosome A site against the offset from the A site. (A) Global attribution plot displaying the average contribution to the A site from 500 codons upstream and downstream from the A site. (B) Local attribution plot displaying the average contribution to the A site from 7 codons upstream and downstream from the A site. (C) Local attribution plot as in [8] displaying the average codon-wise contribution to the A site from 5 codons upstream and downstream from the A site. The codons have been clustered by their respective amino acids. Positive values indicate those that contribute to increasing the ribosome dwell time at the A site, and negative values indicate those that decrease the ribosome dwell time at the A site.

of PCC, with no definitive outcome if we consider the MAE. We hypothesize that the fluctuation of performance is due to the increased amount of information, and noise, that is added through the structural neighbors.

Considering that the performance between the variants of the RiboGL model is not highly different, the DirSeq+ variant of the RiboGL was chosen as the proposed model for the added advantage of being able to study the structure edges by means of extracting interpretability metrics from them.

Interpretability Study using the Captum Graph Explainer

The Captum Graph Explainer (section 3.5) was used to study all the genes in the test set of the mouse liver dataset using predictions made by the RiboGL model. The perturbation of the peaks in the true ribosome profile of these genes was plotted to analyze on a global scale the effect of the different codons in the neighborhood of the ribosome A site. In **Figure 2 (A)**, the distance of the codons from the ribosome A site was plotted against their corresponding mean attributions. It can be noticed that the attributions were mainly from the immediate neighborhood of the A site. We can see from the zoomed-in version of this (**Figure 2 (B)**), that the codons at the E, P, and A sites of the ribosome contribute the most to the prediction at the A site. The codons upstream and downstream from the A site also contribute highly to the prediction at the A site. This figure explains the distance-wise codon importance on the global dataset level, but as the effect is averaged out, it is possible that this does not show long-distance codon relationships that could be important for particular genes. In **Figure 2 (C)** we study the relationship between the different codons and the distance from the A site with the mean attribution to the A site, similarly to [8]. This figure was clustered according to the codons that translate to the same amino acid. Negative values indicate that these reduce the dwell times, and positive values indicate those that increase the dwell times at the A site. For example, the serine coding codon TCG at position -3 is shown to reduce the dwell time at the A site, in that same sense, the cysteine coding TGC codon at the +4 position is shown to increase the dwell time at the A site.

Edge Interpretability with RiboGL - DirSeq+

As one would expect, **Figure 2** shows that the codons in the immediate neighborhood of the A site have the highest importance in predicting the dwell times at the A site. But as these plots show a global perspective, some gene-specific relationships could have been averaged out. In order to study these gene-specific effects, we analyze individual genes. The contributions of the nodes and edges in individual genes in the mouse liver testing set were obtained using the Captum Graph Explainer applied to the proposed RiboGL - DirSeq+ model. One of the best-performing genes, Mitochondrial Ribosomal Protein L11 ("Mrpl11"), was chosen to explain this analysis. In **Figure 3 (A)**, the predicted and true ribosome profiles of the "Mrpl11" gene (in blue and green respectively), along with the global codon-wise contributions (in orange) to the codon at position 96 have been displayed. The peaks in the ground truth ribosome profile of this gene, which represents a position where the ribosomes are stalling more often, were perturbed. We showcase one of those peaks, the codon at position 96 where we can notice that the codons in the local neighborhood have a very high contribution to the dwell time at that position, but in addition to them, there are several distant codons, such as position 48, that also have a high contribution.

In **Figure 3 (B)**, we check the attributions for the same example using a graph perspective to investigate it further. We notice that the distant codon at position 48 which is connected

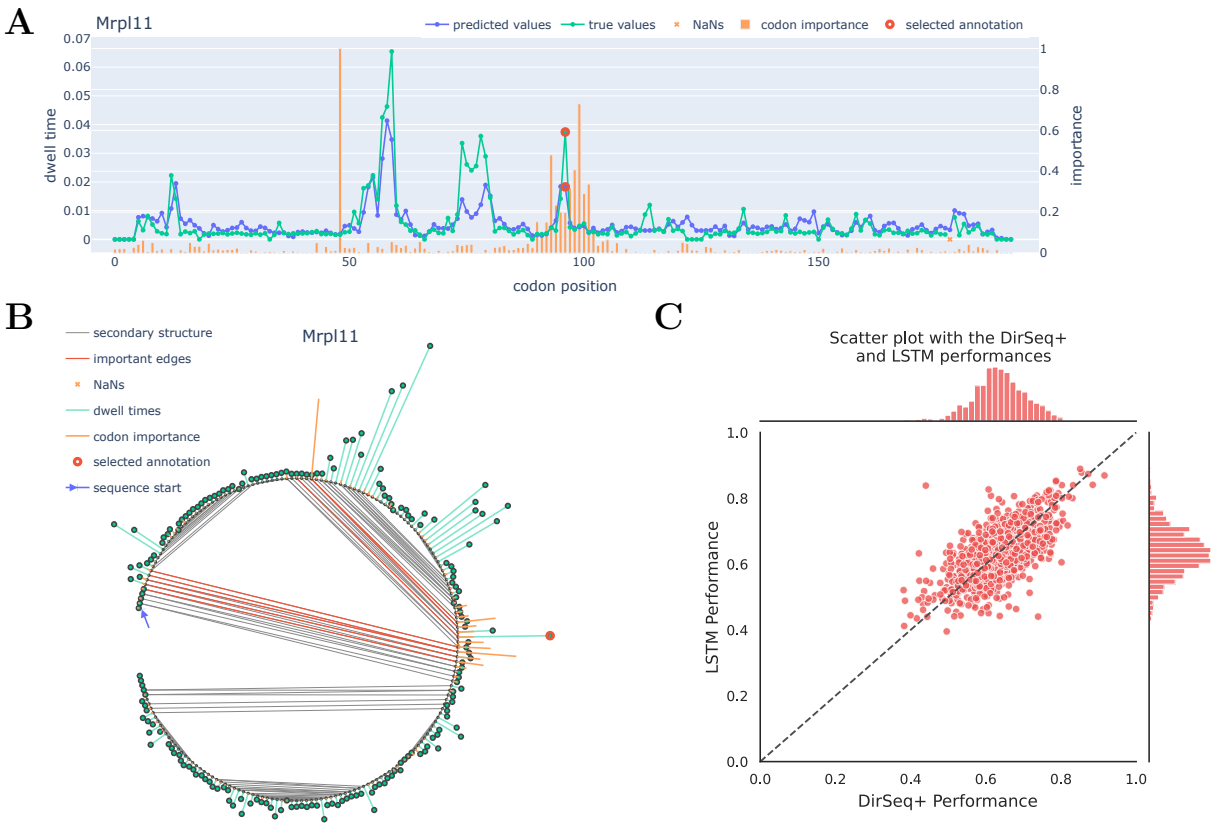

Fig. 3. Interpretability analysis using the Captum Graph Explainer built on the DirSeq+ model on the "Mrpl11" gene, and comparison of the DirSeq+ and LSTM model performances. (A) True and predicted ribosome profiles on the "Mrpl11" gene using the RiboGL - DirSeq+ model. The codon at position 96 was perturbed using the Captum Graph Explainer, and the node attributions for this position have been displayed. (B) True ribosome profile on the "Mrpl11" gene using the RiboGL - DirSeq+ model. The codon at position 96 was perturbed using the Captum Graph Explainer, and all the node and edge attributions for this position have been displayed. The selected codon is the 96 codon ("ATC"), and the long-distance attribution is observed from the codon at position 48 ("TTC"). (C) Scatter plot of the performances of the DirSeq+ model with the LSTM (RiboGL - LSTM), on the mouse liver testing set.

to the codon at position 96, has a high contribution, this could potentially be because they are connected by structure edges. This contribution of the structure edges has been highlighted in the figure. This is an important advantage of the DirSeq+ model, as it allows us to understand and model long-range codon interactions and their contributions through the inclusion of structure edges.

RiboGL and LSTM Performance Analysis

We further investigate the performances of the full RiboGL model with that of the RiboGL-LSTM model. A scatter plot of the performances on the testing set samples for both of these models has been displayed in **Figure 3 (C)** (0.66 PCC). We can notice from this plot that the distribution is mostly on the diagonal, showing that these models have similar performances on all of the sequences. However, the addition of the Graph Processing Block on top of the LSTM allows us to incorporate the structure edges and conduct interpretability studies on them.

Limitations

The proposed RiboGL - DirSeq+ model greatly outperforms the state-of-the-art RiboMIMO model and lays the foundations for modeling ribo-seq data as a graph. However, while the example outlined in the discussion section 4.4 showcases the importance of the structure edges, we did not get a definitive result regarding the performance. We highlight that the information coming from structural neighbors might affect only specific genes or codons, and therefore be lost when averaging the metrics across the whole dataset. Moreover, the results could be highly dependent on the characteristics of the data, such as the coverage distribution (see A.1). Investigating additional datasets, such as bigger datasets belonging to other species, could help clarify the importance of structural neighbors in improving density prediction. Lastly, tools such as "Input X Gradient" should be interpreted with care by taking into consideration their reliability [26]. As a future work, we plan to implement also approaches to increase the robustness of the results, for example by smoothing interpretations [27].

Conclusion

RiboGL is the first graph-based approach applied to predicting full-length ribosome densities. We showcase the importance of using the physical and measurable aspects of mRNA such as the predicted secondary structure, and how we can process

this in terms of a graph. We improved the existing state-of-the-art RNN [8] and combined it with a GNN to make it able to process both the mRNA sequence and secondary structure graph features. We also showed how the graph-based explainability method can be used to study individual genes to understand how node and edge attributions affect the prediction at the A site. In the future, we would like to explore the relationship of ribosome density values with other aspects of the translation, such as the structure of the nascent polypeptide, that can be also modeled by a GNN.

References

1. Nahum Sonenberg and Alan Hinnebusch. Regulation of translation initiation in eukaryotes: Mechanisms and biological targets. *Cell*, 136:731–45, 03 2009.
2. Hani Goodarzi, Hoang Bui Nguyen, Steven Zhang, Brian Dill, Henrik Molina, and Sohail Tavazoie. Modulated expression of specific trnas drives gene expression and cancer progression. *Cell*, 165:1416–1427, 06 2016.
3. Tzu-Yu Liu and Yun S. Song. Prediction of ribosome footprint profile shapes from transcript sequences. *Bioinform.*, 32(12):183–191, 2016.
4. Patrick BF O'Connor, Dmitry E Andreev, and Pavel V Baranov. Comparative survey of the relative impact of mrna features on local ribosome profiling read density. *Nature communications*, 7(1):12915, 2016.
5. Sai Zhang, Hailin Hu, Jingtian Zhou, Xuan He, Tao Jiang, and Jianyang Zeng. Analysis of ribosome stalling and translation elongation dynamics by deep learning. *Cell Systems*, 5(3):212–220.e6, 2017.
6. Robert Tunney, Nicholas J McGlincy, Monica E Graham, Nicki Naddaf, Lior Pachter, and Liana F Lareau. Accurate design of translational output by a neural network model of ribosome distribution. *Nature structural & molecular biology*, 25(7):577–582, 2018.
7. Hailin Hu, Xianggen Liu, An Xiao, Yangyang Li, Chengdong Zhang, Tao Jiang, Dan Zhao, Sen Song, and Jianyang Zeng. Riboexp: an interpretable reinforcement learning framework for ribosome density modeling. *Briefings Bioinform.*, 22(5), 2021.
8. Tingzhong Tian, Shuya Li, Peng Lang, Dan Zhao, and Jianyang Zeng. Full-length ribosome density prediction by a multi-input and multi-output model. *PLoS Comput. Biol.*, 17(3), 2021.
9. Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.
10. Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.
11. Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael Bronstein. Edge directionality improves learning on heterophilic graphs. 2023.
12. Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. 2021.
13. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
14. Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3145–3153. JMLR.org, 2017.
15. Ronny Lorenz, Stephan Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter Stadler, and Ivo Hofacker. Viennarna package 2.0. *Algorithms for molecular biology : AMB*, 6:26, 11 2011.
16. Florian Atger, Cédric Gobet, Julien Marquis, Eva Martin, Jingkui Wang, Benjamin Weger, Grégory Lefebvre, Patrick Descombes, Felix Naef, and Frédéric Gachon. Circadian and feeding rhythms differentially affect rhythmic mrna transcription and translation in mouse liver. *Proceedings of the National Academy of Sciences*, 112(47):E6579–E6588, 2015.
17. Cedric Gobet, Benjamin Weger, Julien Marquis, Eva Martin, Nagammal Neelagandan, Frédéric Gachon, and Felix Naef. Robust landscapes of ribosome dwell times and aminoacyl-trnas in response to nutrient stress in liver. *Proceedings of the National Academy of Sciences*, 117:201918145, 04 2020.
18. Michael Zuker and Patrick Stiegler. Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133–148, 1981.
19. Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2017.
20. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. 2018.
21. Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5782–5799, 2022.
22. Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. 2018.
23. Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. 2021.
24. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.
25. Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch. 2020.
26. David Alvarez-Melis and Tommi S. Jaakkola. On the robustness of interpretability methods. In *WHI 2018*, June 2018.
27. Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.
28. Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2019.

Competing interests

No competing interest is declared.

Author contributions statement

M.V.N. and F.C. conceived and conducted the experiments, M.V.N., F.C. and C.G. analyzed the results. M.V.N., F.C., and C.G. wrote the manuscript, P.V. and F.N. reviewed the manuscript.

Acknowledgments

The authors thank the Swiss National Science Foundation (SNSF) for funding this study (SNSF: # 205884). Additionally, we thank Ali Hariri, Maria Boulougouri, Anaïs Haget, and David Neill Asanza for their valuable insights and discussions regarding the project.

Appendix

Preprocessing

The following steps were conducted after obtaining the pre-processed reads from the Ribo-DT pipeline:

1. **Gene-wise Normalization:** The ribo-seq experiments for this dataset were conducted multiple times to obtain 84 replicates. The codon-level ribosome read counts for these individual replicate experiments were averaged within each gene. Only the longest transcript for every gene was chosen, and the others were removed.
2. **Merging:** The gene-wise normalized ribo-seq replicates were merged to obtain one annotation of ribosome densities per gene.
3. **Annotation:** The codons in all the gene sequences were assigned their respective gene-wise read count values and normalized by the average count. The ranges of these counts were reduced by applying the log1p function.
4. **Pruning:** Multiple threshold conditions were applied while choosing to keep a gene in the final dataset. Those conditions were:
   - Long Sequence of NaNs: All of the gene sequences were traversed and if there were contiguous stretches of zero counts longer than 30 codons, these were converted into NaNs. Once the zeros were converted to NaNs, if those genes had more than 5% of their codons annotated with NaNs, they were removed from the dataset.
   - Long Sequence of Zeros: Genes that had a contiguous sequence of zero count values greater than 20 codons in length were removed from the dataset.
   - Percentage of Sequence Annotated (Coverage): Genes that had less than 30% coverage were removed from the dataset. The coverage was defined by the number of codons annotated with a non-zero and non-NaN count value, divided by the number of codons annotated with a non-zero count value.
5. **Dataset Split:** The resulting dataset consisted of 6,188 genes and this was split into training and testing sets. The genes in the dataset were first ordered in descending order of their coverage. Alternating sequences from the top of this list were added into training, and testing sets until the testing set consisted of 20% of the original dataset. The training set consisted of 4,904 genes and the testing set consisted of 1,284 genes. This kind of train-test split was chosen to maintain the quality of the testing set while reducing the redundancy.

Loss and Metrics

To conduct this training process, a multi-term loss function combining cosine loss and mean absolute error (MAE) was designed. The equations for the loss function have been outlined below.

To evaluate the performance of the models, the Pearson Correlation Coefficient (PCC) was used. The PCC was calculated between the predicted sequence of ribosome densities and the true sequence of ribosome densities.

1. Mean Absolute Error (MAE or L1):

$$MAE = \sum_{i=1}^{D} |x_i - y_i| \tag{5}$$

2. Cosine Loss (CL):

$$CL = 1 - \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6}$$

3. RiboGL Loss

$$Loss = CL + MAE \tag{7}$$

4. Pearson Correlation Coefficient (PCC):

$$PCC = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{8}$$

Model Hyperparameters

The RiboGL model was trained for 300 epochs with an early stopping patience of 20 epochs. The AdamW [28] optimizer was used with a learning rate of $1e-2$, which was reduced by a factor of 0.1 every 10 epochs of the loss not decreasing. A batch size of 2 was used for the training process.

Bootstrapping

In order to obtain the mean performance of a model, along with its confidence intervals, we conducted bootstrapping on the testing set. The predictions on the genes in the testing set were sampled with replacement 1,000 times to obtain 1,000 sets of performances. These were used to obtain the mean and standard deviation of the performance of the model. Additionally, to obtain the 95% confidence intervals, the value of the standard deviation was multiplied by 1.96. The final performance is reported as (mean ± 95% confidence interval)

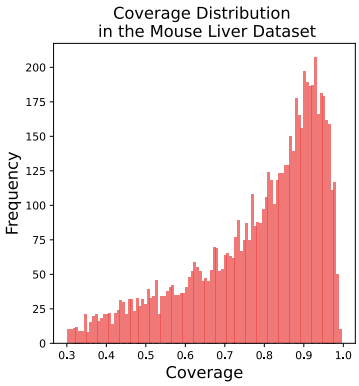

Fig. A.1. Coverage distribution of the genes in the mouse liver dataset

Method	Features	Label Corr.	Out	Predictor
riboShape [3]	ctx	denoise	cdn	LM
RUST [4]	ctx	quant	cdn	CS
ROSE [5]	ctx	quant	cdn	CNN
Iχnos [6]	ctx, fold	norm	cdn	FNN
Riboexp [7]	ctx, fold	norm	cdn	RNN
RiboMIMO [8]	seq, fold	norm, quant	seq	RNN
RiboGL (ours)	seq, graph	norm	seq	GNN+RNN

Table A.1. Summary of ribosome density modeling approaches. Features: codon/nucleotide/aminoacid of the A-site context (ctx) or the full sequence (seq), mRNA folding energies (fold) and secondary structure (graph).
Label Correction: denoising (denoise), quantization (quant), and normalization, e.g., dividing by average transcript density (norm).
Output: at codon (cdn) or sequence (seq) level.
Predictor: Codon-wise Statistics (CS), Linear Model (LM), and Feedforward, Convolutional and Recurrent Neural Network (FNN, CNN, and RNN, respectively).